Learning Content Integration into an Electronic Textbook for Introductory Programming

Jordan Barria-Pineda
University of Pittsburgh
jab464@pitt.edu

Kamil Akhuseyinoglu
University of Pittsburgh
kaa108@pitt.edu

Peter Brusilovsky
University of Pittsburgh
peterb@pitt.edu

ABSTRACT

In this paper, we describe an initial attempt for integrating multiple smart learning content from different providers into an e-reader platform. We present an integration example for the case of an introductory programming textbook. We show that it is possible to embed different related external content through standardized protocols, including LTI (Learning Tools Interoperability).

KEYWORDS

interoperability, LTI, smart learning content, electronic textbook

ACM Reference Format:
Jordan Barria-Pineda, Kamil Akhuseyinoglu, and Peter Brusilovsky. 2019. Learning Content Integration into an Electronic Textbook for Introductory Programming. In *Proceedings of SPLICE'19*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

During the last decades, numerous research teams around the world have been working on creating novel digital learning tools to help students to learn different domains. These educational tools differ from each other by the different levels of complexity and specificity of their learning objectives, as well as by the interaction features they present to learners. Usually, the benefits of these tools on students' learning are evaluated in isolation, i.e. learners have access to only one type of learning content throughout each research study, without investigating the potential effect of providing access to several types of learning activities at once. Only a few studies have explored the integration of different learning content providers [3], which suggest that this has a positive effect on students learning.

Notwithstanding the above, textbooks in their traditional form remain as one of the most common means of instruction. The use of e-readers for accessing text-based learning content has become widespread, but digital textbooks usually lack interactive features which could make the learning a more engaging process. Some attempts of integrating interactive learning activities within a static textbook have been done in recent years. For example, in [5] an introductory Python textbook was enhanced with interactive content like Parsons problems and multiple- choice questions. In this case, the embedded content was created by the same producers of the textbook. The drawback of this approach is that the content generation is time-consuming task and the learning activities are not reusable by other interested parties.

In this work, we integrated multiple smart learning contents into an interactive e-textbook using standardized protocols including Learning Tool Interoperability (LTI) [9]. Smart learning content integration was visioned by the ACM ITiCSE working group [2], and we show the integration and reuse of smart learning contents developed and served by various research institutions.

2 BACKGROUND

Reading Mirror is an e-reader platform developed by the University of Pittsburgh that enables instructors to upload course reading material by following the syllabus structure. Thus, students have online access to read the course weekly readings and to answer related quizzes, all in the same system. Additionally, learners can visually check their progress and compare it to the rest of the class, as a way of increasing their awareness about their peers' study behaviors. Although e-readers normally rely on *pdf* document rendering, this system delivers the material by using HTML5, which makes it more versatile when it comes to embedding external directly on the documents.

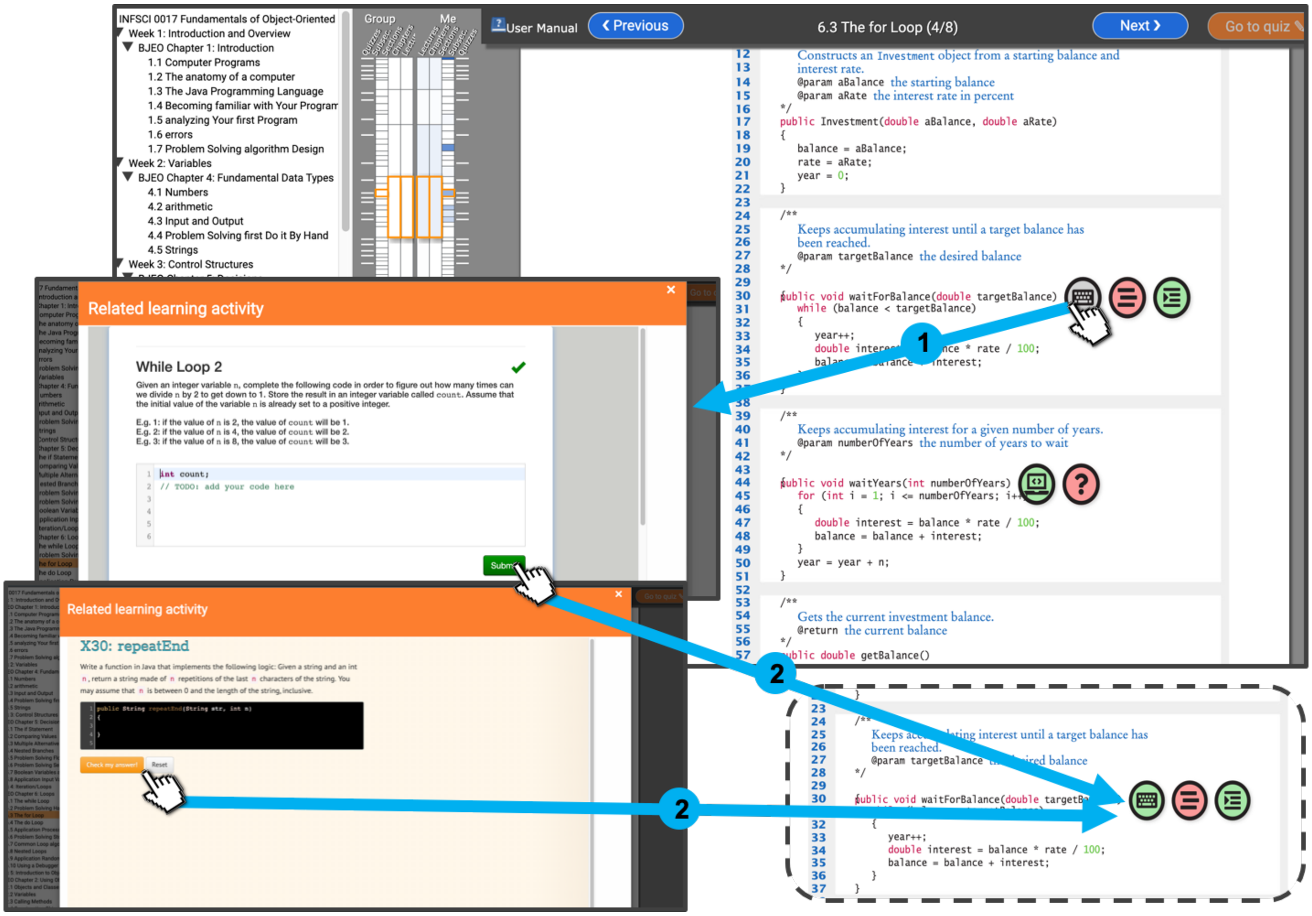
To be able to integrate multiple learning contents, we leverage the usage of standardized protocols such as ADAPT2 proprietary content invocation protocol [1] which has been developed by our research team and LTI [9] which has been developed by IMS Global Learning Consortium. LTI let Learning Management Systems (LMS) access third-party educational tools securely. Similarly, ADAPT2 protocol enables learning systems to access existing educational content. Different from LTI, ADAPT2 protocol is focused mainly on adaptive learning contents and learner actions generated by these contents are reported back to a dedicated student modeling service.

In LTI protocol, systems which access learning applications through the LTI protocol are called *Tool consumers* and the LTI-complaint learning applications are called *Tools*. In this work, we enhanced Reading Mirror with LTI tool consumer capabilities and integrated an LTI-compatible online practice system along with other learning contents integrated through ADAPT2 protocol. The integration process is described in the next section.

3 INTEGRATING SMART LEARNING CONTENT INTO READING MIRROR

The process of embedding learning content from different providers is possible given the use of multiple *interoperability protocols*. To prove the possible outcome of such integration, we have integrated CodeWorkout [4] which is an LTI-compatible small-scale programming assignments tool. In addition to LTI protocol integration, students also have access to other learning activities which are

SPLICE'19, August 11th, 2019, Toronto, ON, Canada
© 2019 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



INFSCI 0017 Fundamentals of Object-Oriented
- Week 1: Introduction and Overview
  - BJEO Chapter 1: Introduction
    - 1.1 Computer Programs
    - 1.2 The anatomy of a computer
    - 1.3 The Java Programming Language
    - 1.4 Becoming familiar with Your Program
    - 1.5 analyzing Your first Program
    - 1.6 errors
    - 1.7 Problem Solving algorithm Design
- Week 2: Variables
  - BJEO Chapter 4: Fundamental Data Types
    - 4.1 Numbers
    - 4.2 arithmetic
    - 4.3 Input and Output
    - 4.4 Problem Solving first Do it By Hand
    - 4.5 Strings
- Week 3: Control Structures

```
   Constructs an Investment object from a starting balance and
   interest rate.
   @param aBalance the starting balance
   @param aRate the interest rate in percent
*/
public Investment(double aBalance, double aRate)
{
   balance = aBalance;
   rate = aRate;
   year = 0;
}

/**
   Keeps accumulating interest until a target balance has
   been reached.
   @param targetBalance the desired balance
*/
public void waitForBalance(double targetBalance)
   while (balance < targetBalance)
   {
      year++;
      double intere      nce * rate / 100;
      bala               nterest;

}

/**
   Keeps accumulating interest for a given number of years.
   @param numberOfYears the number of years to wait
*/
public void waitYears(int numberOfYears)
   for (int i = 1; i <= numberOfYears; i++
   {
      double interest = balance * rate / 100;
      balance = balance + interest;
   }
   year = year + n;
}

/**
   Gets the current investment balance.
   @return the current balance
*/
ublic double getBalance()
```

Related learning activity

While Loop 2 ✓

Given an integer variable n, complete the following code in order to figure out how many times can we divide n by 2 to get down to 1. Store the result in an integer variable called count. Assume that the initial value of the variable n is already set to a positive integer.

E.g. 1: if the value of n is 2, the value of count will be 1.
E.g. 2: if the value of n is 4, the value of count will be 2.
E.g. 3: if the value of n is 8, the value of count will be 3.

```
int count;
// TODO: add your code here
```

Submit

Related learning activity

X30: repeatEnd

Write a function in Java that implements the following logic: Given a string and an int n, return a string made of n repetitions of the last n characters of the string. You may assume that n is between 0 and the length of the string, inclusive.

```
public String repeatEnd(String str, int n)
{

}
```

Check my answer! Reset

Figure 1: Smart learning content integration workflow reflected in the student's actions on the Reading Mirror platform

integrated through ADAPT2 protocol such as program construction examples and challenges [6], semantic problems [7] and PCRS coding exercises [10]. The integrated tools are physically served at servers located in different countries and they provide multiple learning contents. These learning contents can be accessed by a unique content URL (content link) using either LTI or ADAPT2 protocol.

In development process, the Reading Mirror is kept isolated from the details of integration protocols. We reuse our *adapter pattern* approach for LTI-consumer service developed in [8] to serve CodeWorkout exercises in Reading Mirror. This service generates appropriate LTI launch requests given the student's session data (e.g. username, course, etc.). Similarly, other content types are integrated through ADAPT2 protocol by sending HTTP GET requests to the content providers. Both protocols eliminate tedious log-in procedures and provide a smooth learning experience.

For explanatory purposes, we use Cay Horstmann's *Big Java Book* as a source for embedding smart learning content. The workflow of interrelated actions that makes this integration possible is depicted in the Fig. 1, and explained as follows:

(1) Each learning content is embedded into the proper textbook section using content links. The embedded learning activity is added to the HTML5 page as a circular icon (which is unique for each content provider). When the learner clicks on the respective content icon (see arrow 1 in Fig. 1), the activity's expanded URL is loaded in a modal window. Here, learners can start interacting with the learning content.

(2) The learning events generated at integrated content are reported back to user modeling service either through the LTI grade callback service developed in [8] (for LTI protocol) or using ADAPT2 protocol. Following this, the interface is updated to provide immediate feedback to the student. This is reflected through the color of the activity's icon: it turns green if the submission was correct or red if it was incorrect; otherwise, it keeps gray if it was not attempted (see 2 in Fig. 1).

4 CONCLUSION AND FUTURE WORK

In this paper we briefly presented the embedding of several types of interactive learning activities into an existent digital textbook reader by using different learning *interoperability protocols* (LTI and ADAPT2). Currently, the process of associating the learning activities is manual, so we plan to develop an automatic or semi-automatic approach for finding the most appropriate section of the book to embed each content given its conceptual coverage. Finally, it is important to mention that this is an ongoing work, so more work is needed in order to evaluate its usefulness and its effect on students taking a real introductory programming class.

REFERENCES

[1] P. Brusilovsky. Knowledgetree: A distributed architecture for adaptive e-learning. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 104–113. ACM, 2004.

[2] P. Brusilovsky, S. Edwards, A. Kumar, L. Malmi, L. Benotti, D. Buck, P. Ihantola, R. Prince, T. Sirkiä, S. Sosnovsky, et al. Increasing adoption of smart learning content for computer science education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, pages 31–57. ACM, 2014.

[3] P. Brusilovsky, L. Malmi, R. Hosseini, J. Guerra, T. Sirkiä, and K. Pollari-Malmi. An integrated practice system for learning programming in python: design and evaluation. *Research and Practice in Technology Enhanced Learning*, 13(1):18, 2018.

[4] K. Buffardi and S. H. Edwards. Introducing CodeWorkout: An Adaptive and Social Learning Environment. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pages 724–724. ACM, 2014.

[5] B. J. Ericson, M. J. Guzdial, and B. B. Morrison. Analysis of interactive features designed to enhance learning in an ebook. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research,ICER '15*, pages 169–178. ACM, 2015.

[6] R. Hosseini, K. Akhuseyinoglu, A. Petersen, C. D. Schunn, and P. Brusilovsky. Pcex: Interactive program construction examples for learning programming. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, page 5. ACM, 2018.

[7] I.-H. Hsiao, S. Sosnovsky, and P. Brusilovsky. Guiding students to the right questions: adaptive navigation support in an e-learning system for java programming. *Journal of Computer Assisted Learning*, 26(4):270–283, 2010.

[8] H. Manzoor, K. Akhuseyinoglu, C. Shaffer, and P. Brusilovsky. Opendsa/mastery grids exercise interchange. In *Proceedings of the Fourth SPLICE Workshop co-located with 50th ACM Technical Symposium on Computer Science Education (SIGSCE 2019), Minneapolis, MN, USA, February 27 - March 2, 2019.*

[9] C. Severance, T. Hanss, and J. Hardin. IMS Learning Tools Interoperability: Enabling a Mash-up Approach to Teaching and Learning Tools. *Technology, Instruction, Cognition and Learning*, 7(3-4):245–262, 2010.

[10] D. Zingaro, Y. Cherenkova, O. Karpova, and A. Petersen. Facilitating code-writing in pi classes. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 585–590. ACM, 2013.